Combining Multi-Level Data to Assess System Reliability and Allocate Resources Optimally: Bayesian Methods and Computation

Todd L. Graves and Michael Hamada
Statistical Sciences
Los Alamos National Laboratory
October 19, 2004

Abstract

Good estimates of the reliability of a system make use of test data and expert knowledge at all available levels. Furthermore, by integrating all these information sources, one can determine how best to allocate scarce testing resources to reduce uncertainty. Both of these goals are facilitated by modern Bayesian computational methods and the YADAS software in particular. We apply these tools to classical examples that were previously solvable only through the use of ingenious approximations, and use genetic algorithms to guide resource allocation.

Key Words: Reliability Block Diagram, System-Level, Subsystem-Level and Component-Level Data, Bayesian Methods, Genetic Algorithm, Resource Allocation, MCMC, Metropolis-Hastings Algorithm.

1 Introduction

This article is concerned with assessing the reliability of a system by combining all data and information at whatever level they are available, whether they are at the component, subsystem or system level. Much of the reliability literature (Cole (1975), Mastran (1976), Mastran and Singpurwalla (1978), Natvig and Eide (1987), Martz, Waller and Fickas (1988), Martz and Waller (1990)) predates the advances made in Bayesian computation in the 1990's and resorts to various approximations. However, today a fully Bayesian method using the Johnson et al. (2003) framework which combines all available multi-level level data and information can be implemented using Markov chain Monte Carlo (MCMC).

While it is possible to use BUGS (Spiegelhalter et al. (2000)) to carry out reliability assessments, BUGS code needs to be tailored specifically to the structure of a system (and what data and information are available). In Section 2, we present YADAS (Yet Another Data Analysis System), a recently developed statistical modeling environment, and show how it can easily handle such system reliability assessments. We illustrate YADAS's capability with reliability assessments of an air-to-air heat-seeking missile system and a low-pressure coolant injection system first considered by Martz et al. (1988) and Martz and Waller (1990), respectively, in Section 3.

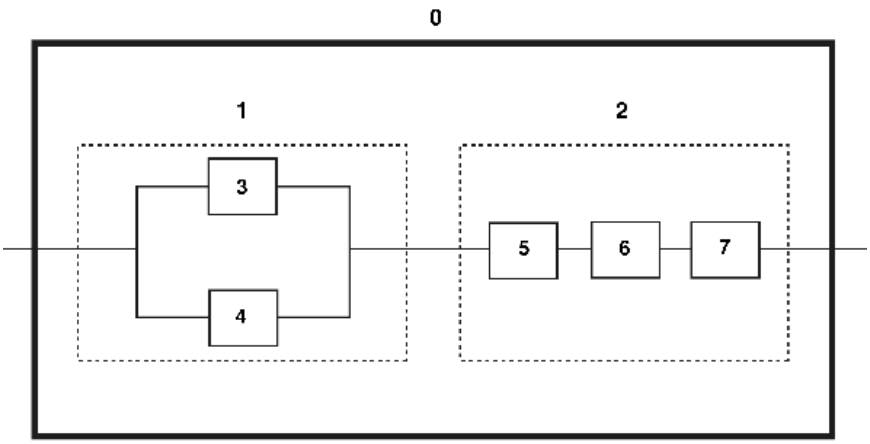

Figure 1: Simplified System Reliability Block Diagram.

Once multi-level data and information can be analyzed, the question arises of what additional tests should be done when new funding becomes available. That is, what tests will reduce the system reliability uncertainty the most? In Section 4, we show how a genetic algorithm using a pre-posterior based criterion can address this resource allocation question. This will also be illustrated with the simplified system depicted in Figure 1.

To combine multi-level data for system reliability assessment, we use the Johnson et al. (2003) framework. We will introduce the framework's notation and models by considering the reliability block diagram of a simplified system given in Figure 1. First, components, subsystems and the system are referred to as nodes. In this example, the system is node 0 which consists of two subsystems (nodes 1 and 2) in series. The first subsystem consists of two components in parallel (nodes 3 and 4) and the second subsystem consists of three components in series (nodes 5, 6 and 7).

We begin by considering the binomial data model when data are available at a node. At the ith node, there are x_i successes in n_i trials with reliability π_i. If node i is a subsystem or the full system (i.e., not a component), then π_i is expressed in terms of the component reliabilities. For the simplified system, the subsystem reliabilities are expressed as $\pi_1 = 1-(1-\pi_3)(1-\pi_4)$ and $\pi_2 = \pi_5\pi_6\pi_7$ and the system reliability is expressed as $\pi_0 = \pi_1\pi_2 = \{(1-(1-\pi_3)(1-\pi_4))\pi_5\pi_6\pi_7\}$.

Next, we consider priors for node reliabilities. For components, we use beta priors in terms of a best guess for reliability $\tilde{p}_i$ and a precision ν_i which acts like an effective sample sample. That is, if the ith node is a component, then $\pi_i \sim Beta(\nu_i\tilde{p}_i, \nu_i(1-\tilde{p}_i))$. If no information is available, the Jeffreys' prior $Beta(1\times 0.5, 1\times(1-0.5))$ or a uniform prior $Beta(2\times 0.5, 2\times(1-0.5))$ can be used.

If information is available at their levels, subsystem and system priors are treated as binomial data as

Table 1: Data for Simplified System

Node	Data	$\tilde{p}$
0	15/20	0.8
1		0.9
2	10/10	0.9
3	34/40	0.9
4	47/50	0.9
5	3/5	0.95
6	8/8	0.95
7	16/17	0.95

follows. If $\tilde{p}_i$ is the estimated reliability and its precision is ν_i, then the likelihood is proportional to

$$\pi_i^{\nu_i \tilde{p}_i} (1 - \pi_i)^{\nu_i (1 - \tilde{p}_i)}.$$

As discussed above the subsystem or system reliability π_i is expressed in terms of the component reliabilities. Note that treating this information as data differs from treating it as a prior by changing the exponents of π_i and $(1 - \pi_i)$ by one.

There is a variety of models that might be employed for the ν_i. The ν_i might be treated as constants when they are really thought to be effective sample sizes. On the other hand, they might be described by a distribution, such as $\nu_i \sim Gamma(a_\nu, b_\nu)$. This allows expert knowledge to be downweighted if it is inconsistent with the data or with itself. The ν_i may also be grouped into prior classes so that the ν_i within a group are equal. Now consider the data and prior information for the simplified system given in Table 1. Note that no precisions ν_i are provided so that a prior distribution needs to be specified. For illustration, we consider the same precision ν, i.e., $\nu_i = \nu$, and take the prior for ν to be:

$$\nu \sim Gamma(a_\nu = 5, b_\nu = 1).$$

The resulting priors for the node reliabilities and ν are displayed as dotted lines in Figures 2 and 3. Combining these priors with the node data using MCMC result in the posteriors displayed as the solid lines in Figure 2 and Figure 3. From these results, the 90% credible interval for the system (node 0) reliability is calculated as (0.697, 0.861) whose length is 0.164. Note that even though there is no data for the first subsystem (node 1), the system data (node 0) and the component data (nodes 3 and 4), dramatically improve what we know about the first subsystem reliability. The addition of the data does not much change ν except that ν is somewhat larger than indicated by the prior.

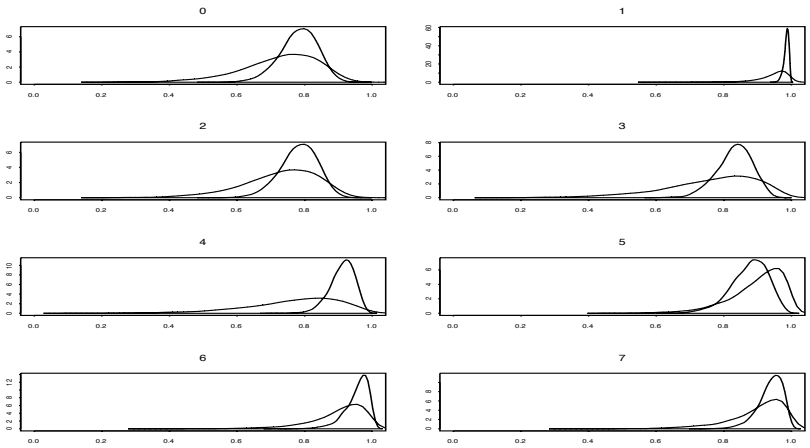

Figure 2: Plot of Simplified System Reliability Priors (flatter curves) and Posteriors (peaked curves) for Nodes 0-7.

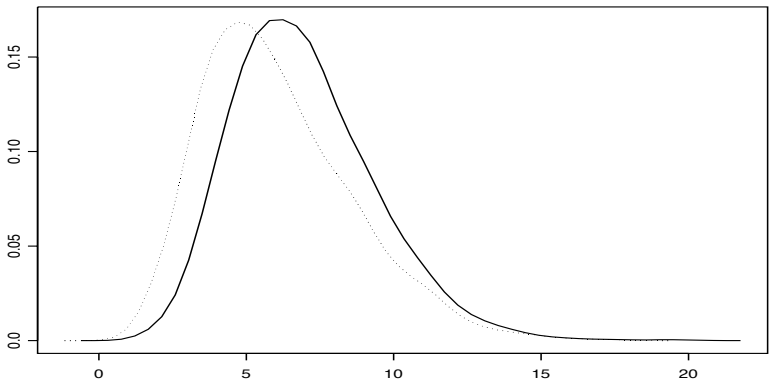

Figure 3: Plot of Simplified System ν Prior (dotted line) and Posterior (solid line).

2 YADAS: a Statistical Modeling Environment

A full Bayesian analysis of the model described above is nontrivial. The posterior distribution is not analytically tractable; although the data and expert opinion likelihood looks like

$$\prod_i \pi_i^{A_i}(1-\pi_i)^{B_i} \tag{1}$$

for some choices of A_i and B_i, this does not imply a beta distribution because of the functional relationships between the π_i. Consequently, an analysis requires an implementation of a Markov chain Monte Carlo algorithm to estimate the unknown parameters (and the Gibbs sampler using exact samples from conditional distributions is not practical either). The problem is novel enough that existing software packages such as BUGS (Spiegelhalter et al. 2000) are not overly useful in solving it. For us, using this model was straightforward due to the existence of the YADAS system (Graves, 2003a,b). Some of YADAS's goals are

- saving the user from having to work out algebra before writing code,
- providing commonly used components that facilitate constructing new models,
- preventing explosion of repetitious code that is hard to debug (previous BUGS implementations of related problems have suffered from this difficulty), and
- making it easy to make changes to a model after initial coding.

We now discuss some of the elements of the YADAS design that help it achieve these goals.

2.1 Brief discussion of YADAS architecture

YADAS is a free, open–source system distributed, along with a tutorial, examples, and supporting papers, at `yadas.lanl.gov`. It is written in Java, an object–oriented language that encourages abstraction and general solutions. The fundamental concepts of object–oriented programming are outside the scope of this paper, but we will discuss three key abstractions of YADAS: the Parameter, the Bond, and the Update.

"Parameters" in YADAS are the quantities whose posterior distribution we are trying to estimate. They keep track of their values as the algorithm proceeds, and each parameter knows a default method for taking a step in the MCMC algorithm (normally variable-at-a-time Gaussian random walk Metropolis). In the system reliability example, the key parameter is the vector of π_i's. This vector contains one element for each component; the π_i's for the other nodes are functions of the component π_i's and are handled in other ways.

"Bonds" are the terms in the unnormalized posterior distribution (some corresponding to the prior, others to the likelihood). These represent relationships between parameters, data, and constants. The most common way of defining a bond is to use the "basic bond", whose definition consists of a subset of the parameters, a density function (e.g. Gaussian, Gamma, Binomial and so forth), and a vector of functions for converting the parameters into arguments for the density functions. For example, the univariate Gaussian

density function requires a vector of data points, a vector of means, and a vector of standard deviations. In the system reliability example, a good example of a bond is the one that handles all the binomial test data at all levels. Numbers of successes and sample sizes are data, and one also needs to calculate the success probabilities for each node (component, subsystem, and the whole system) as functions of the component success probabilities.

Finally, "updates" combine to define the MCMC algorithm used to estimate the posterior distribution. The most basic kind of update is the kind that applies to each parameter and defines a variable-at-a-time, Gaussian random walk Metropolis move. Other updates attempt to adjust multiple parameters simultaneously. The updates used in the system reliability problem are not too different from the simple types, but since the parameters are probabilities that can be close to one, we perform Gaussian random walk Metropolis-Hastings moves on the logits of the parameters.

2.2 YADAS for system reliability

Another strength of YADAS derives from its open source distribution: there are no barriers to adding problem–specific code to facilitate analysis of particular problems. In the system reliability example, we created a new package called ReliableSystem whose two main responsibilities were reading in the system structure from an input file and computing the subsystem success probabilities as functions of the component success probabilities. This has enabled us to analyze several systems using the same analysis code, and the input file structure is simple enough to be quite easy to debug.

3 Examples

In this section we consider two substantive examples from the literature (Martz et al. (1988) and Martz and Waller (1990)) to demonstrate the capability of YADAS.

3.1 Series Systems Example

Martz et al. (1988) considered the reliability of a certain air-to-air heat-seeking missile system consisting of five subsystems in series each consisting of multiple components themselves combined in series as depicted in Figure 4. The data and prior information that Martz et al. (1988) used are presented in Table 2 as (successes/trials) and best guesses $\tilde{p}$ and precisions ν.

To compare with Martz et al. (1988), we treat the precisions as constants and obtain the posterior node reliabilities using YADAS. The posterior node reliabilities are displayed in Figure 5 as solid lines; the Martz et al. (1988) results are displayed as dotted lines. The median (0.50 quantile) and 90% credible intervals (0.05, 0.95 quantiles) for the system and subsystem posterior reliabilities from the full Bayes and Martz et al. (1988) methods are given in Table 3.

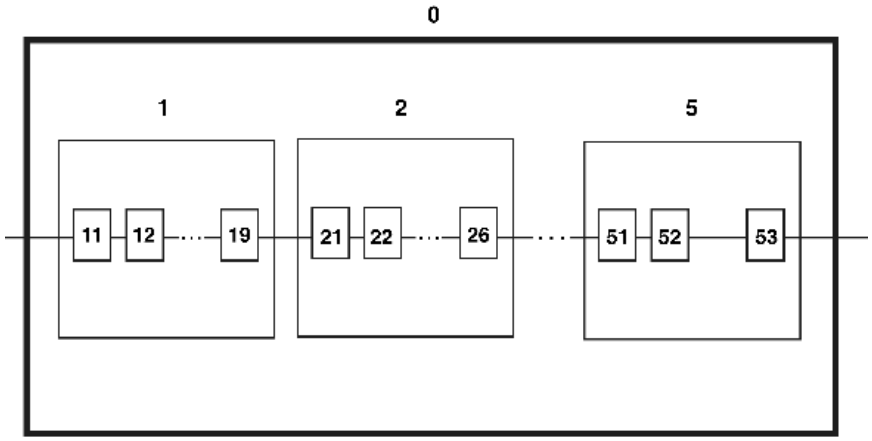

Figure 4: Series System Example Reliability Block Diagram.

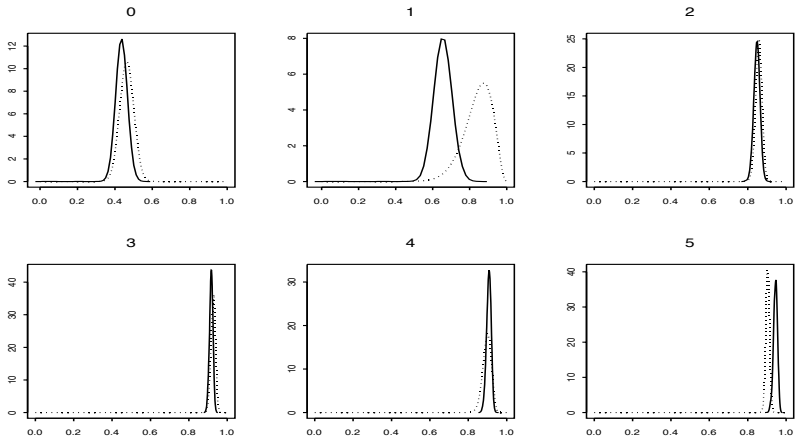

Figure 5: Plot of Series System Example Reliability Posteriors (dotted from Martz et al. (1988) paper, solid full Bayesian).

Table 2: Data for Series System Example

Node	Data	$\tilde{p}$	ν
0		116/267	267
1	8/8		
2	7/8		
3	191/205	258/271	271
4		56/68	68
5			
11	30/30	0.5	1
12	80/80	0.5	1
13	39/40	0.5	1
14	30/30	0.5	1
15	90/90	846/848	848
16	10/10	0.5	1
17	29/30	0.5	1
18	20/20	0.5	1
19	5/5	0.5	1
21	50/50	399/401	401
22	50/50	278/302	302
23	99/100	1098/1102	1102
24	23/25	654/689	689
25	50/50	299/301	301
26	55/55	348/352	352
31	129/130	246/250	250
32	130/130	245/250	250
33	129/130	247/250	250
34	129/130	272/276	276
35	130/130	357/360	360
36	247/250	254/257	257
37	129/130	250/252	252
38	249/250	250/252	252
39	330/330	341/352	352
41		797/802	802
42		796/802	802
43		794/802	802
44		791/802	802
45		386/402	402
51		1026/1122	1122
52		1087/1092	1092
53		1084/1092	1092

Table 3: Comparison of Posteriors for Series System Example (0.05, 0.5, 0.95 quantiles)

Node	Full Bayes	Martz et al.
0	(0.393, 0.436, 0.479)	(0.403, 0.463, 0.525)
1	(0.588, 0.655, 0.723)	(0.701, 0.851, 0.947)
2	(0.820, 0.848, 0.873)	(0.830, 0.858, 0.883)
3	(0.901, 0.917, 0.931)	(0.908, 0.927, 0.944)
4	(0.886, 0.908, 0.925)	(0.858, 0.898, 0.931)
5	(0.926, 0.945, 0.961)	(0.889, 0.904, 0.918)

Note that there is quite a difference in the subsystem 1 results. The difference in location is due to the fact that the approximations used in Martz et al. (1988) do not use higher–level information (system data) to inform estimates of lower–level parameters (such as subsystem 1 reliability). The expert judgment estimate of system reliability, 116/267 or 0.43, is lower than the data and expert judgment at the lower levels would imply, and the full Bayesian analysis needs to attribute this unreliability to one of the subsystems. Subsystem 1 and in particular component 19 have the sparsest information and are the natural targets. For this reason, the full Bayesian analysis is more useful than the approach of Martz el al. (1988) in evaluating the usefulness of gathering more data at low levels. In practice one would review the information that led to the low system reliability estimate. The full Bayesian analysis could be rerun with random ν's, and this would presumably allocate positive probability to the event that $\tilde{p}_0$ is an underestimate.

3.2 Complex Series/Parallel System Example

Martz and Waller (1990) considered the reliability of a low-pressure coolant injection system, an important safety system in a nuclear-power boiling-water reactor. It consists of twin trains consisting of pumps, valves, heat exchanges and piping whose reliability block diagram is displayed in Figure 6. The data and prior information that Martz and Waller (1990) used are presented in Table 4 as (successes/trials) and best guess $\tilde{p}$ and precision ν.

Like Martz and Waller (1990), we treat the precisions as constants and obtain the posterior node reliabilities using YADAS. The resulting posterior node reliabilities are displayed in Figure 7.

4 Resource Allocation

Once there is a way to analyze data, then test design can be addressed. When additional funding becomes available, the questions of where should the tests be done and how many should be taken arise. In this section, we consider the optimal allocation of additional testing within a fixed budget that results in the least uncertainty of system reliability. We explore this by using the simplified system of Section 1. We must determine how many tests should be performed at the system, subsystem and component level (i.e., nodes 0-7) under a fixed budget for specified costs at each level (system, subsystem, component). We use a genetic

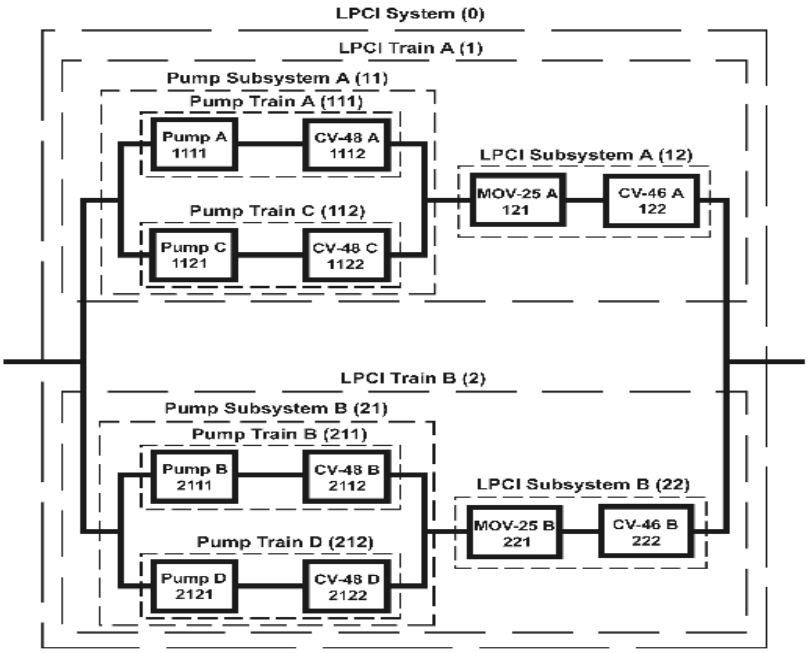

Figure 6: Complex Series/Parallel System Example Reliability Block Diagram.

Table 4: Data for Complex Series/Parallel System Example

Node	Data	$\tilde{p}$	ν
0			
1			
2			
11			
12		242.87/244.66	244.66
111		1.55/1.58	1.58
112		1.55/1.58	1.58
121	80/80	470.13/471.90	471.90
122	39/40	14232.34/14234.12	14234.12
1111	30/30	191.17/191.79	191.79
1112	90/90	14232.34/14234.12	14234.12
1121	10/10	191.17/191.79	191.79
1122	29/30	14232.34/14234.12	14234.12
21			
22		242.87/244.66	244.66
211		1.55/1.58	1.58
212		1.55/1.58	1.58
221	80/80	470.13/471.90	471.90
222	39/40	14232.34/14234.12	14234.12
2111	30/30	191.17/191.79	191.79
2112	90/90	14232.34/14234.12	14234.12
2121	10/10	191.17/191.79	191.79
2122	29/30	14232.34/14234.12	14234.12

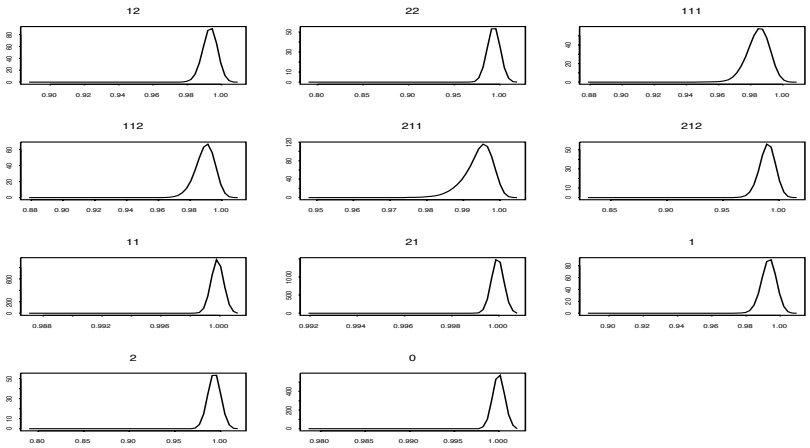

Figure 7: Plot of Complex Series/Parallel System Example Reliability Posteriors.

algorithm (GA) (Goldberg (1989), Michalewicz (1992)) to do the optimization.

Thus, we assume that there is a cost for collecting additional data with higher-level data being more costly than lower-level data. Consider the following costs as an example of the costs for testing at each node. Recall that node 0 is the system, nodes 1 and 2 are subsystems and nodes 3-7 are components:

(0: \$3), (1: \$2), (2: \$2), (3: \$1), (4: \$1), (5: \$1), (6: \$1), (7: \$1).

We evaluate a candidate allocation (i.e., number of tests for each node) using a pre-posterior based criterion as follows. We take a draw from the current joint posterior (based on the current data) of the node reliabilities and draw binomial data according to the candidate allocation. Then we combine these new data with the current data using the same priors to obtain an updated posterior of the node reliabilities. The length of the 90% central credible interval of the system reliability posterior is taken as the uncertainty. This is repeated N times, each with a different draw from the current joint posterior of the node reliabilities. The uncertainty criterion is the 0.90 quantile of the resulting 90% credible interval lengths.

Briefly, we describe how a GA can be used to find a nearly optimal allocation. A GA operates on a "population" of candidate allocations, where a candidate allocation is a vector of node sample sizes. The GA begins by constructing an initial population or generation of M allocations by randomly generating allocations that do not exceed the given fixed budget. The uncertainty criterion for each of these allocations in the initial population is evaluated and the allocations are ranked from smallest to largest, i.e., the best allocation has the smallest criterion in the initial population. The second (subsequent) GA generations are then populated using two genetic operations: crossover and mutation. A crossover is achieved by randomly selecting two parent allocations from the initial (current) generation without replacement with probabilities inversely proportional to their rank among the M allocations in the initial (current) generation. A new allocation is obtained from the two parent allocations node by node by randomly picking one of the two

parents each time and taking its node sample size. The two parent allocations are then returned to the initial (current) population before the next crossover is performed. In this way, an additional M allocations are generated using the crossover operator. The generated allocations are checked to make sure they do not exceed the budget, so that new allocations are generated until there are M such allocations. The uncertainty criterion is then evaluated for each of these new allocations. A mutation of each of the initial (current) M allocations is obtained node by node by first randomly deciding to change the node sample size and if so then randomly perturbing the current sample size. Using mutation, M additional allocations which stay within the budget are generated and the uncertainty criterion for each is evaluated. At this point there are $3M$ allocations. In the next generation, the current population consists of the M best allocations from these $3M$ allocations, i.e., with the smallest uncertainty criterion. The GA is executed for G generations. We implemented the GA for resource allocation in R (R Development Core Team (2004)) which generates the candidate allocations. An allocation is evaluated in R by repeated building YADAS input data files, using YADAS (through the "system" call) to analyze the new and current data and reading the resulting YADAS output files to calculate the uncertainty criterion.

In the simplified system of Section 1.1, the length of the 90% credible interval of system reliability based on the existing data was 0.164. To illustrate the GA for the allocation problem described above, we consider a fixed budget of \$1000. Populations of size $M = 20$ were used to generate $G = 50$ generations. Consequently a total of 2020 ($= 20+40\times 50$) allocations are generated and evaluated. The uncertainty criterion is based on 100 draws from joint posterior of reliabilities obtained in Section 1.1. For a budget of \$1000, what resource allocation yields the most reduction in the uncertainty criterion for system reliability? Based on the GA escribed above, the GA produced the traces presented in Figures 8 and 9 which display the best uncertainty criterion and allocation found during each generation. The uncertainty criterion starts at 0.085 for the initial population and decreases to 0.069 in generation 50 with an allocation of (93, 3, 175, 34, 19, 153, 67, 77) for nodes 0-7. Note that an allocation using the entire budget was not identified because the slight improvement over that found was within the simulation error of the uncertainty criterion. Note that in the best allocation very few subsystem 1 tests (node 1) and subsystem 1 component tests (nodes 3 and 4) are required.

5 Discussion

We have illustrated how to respond to the challenge of integrating all information available at the various levels of a system in order to estimate its reliability. Bayesian models have always been natural for doing this integration, and the computational tools have now caught up to make this practical. The YADAS modeling environment has been very useful in creating general code with easily constructed input files for this purpose; the `ReliableSystem` package, which fits naturally into the YADAS architecture, does the novel work. The code is automatic enough that it can be used inside a genetic algorithm written in R whose purpose is to find near–optimal allocations of resources.

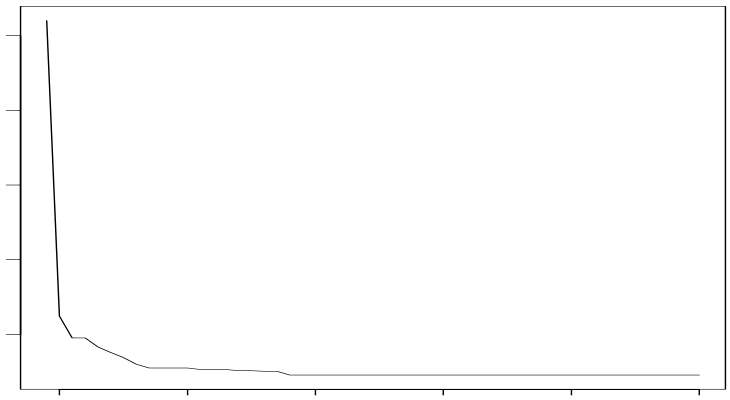

Figure 8: GA Evolution of Best Uncertainty Criterion.

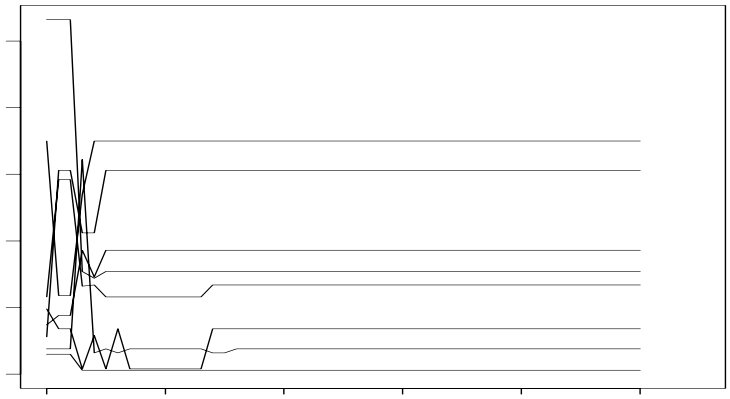

Figure 9: GA Evolution of Best Resource Allocation.

We have discussed the case of binomial test data only, but YADAS is also general enough to handle component and subsystem tests that generate continuous data such as lifetimes, and their distributions can depend on covariates such as age. See Graves and Hamada (2004). Future work includes generalizing the system reliability modeling code to handle common cause failures, by including software to calculate minimal cut sets and using these results to calculate subsystem success probabilities. See Hamada et al. (2004) for a similar analysis.

Acknowledgments

We thank Dee Won for her encouragement of this work. We thank Vivian Romero for her assistance in producing the reliability block diagram figures used in this paper.

References

Chib, S., Greenberg, E. (1995). Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49, 327-335.

Cole, P.V.Z. (1975). A Bayesian reliability assessment of complex systems for binomial sampling. *IEEE Transactions on Reliability*, R-24, 114-117.

Gelfand, A.E., Smith, A.F.M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85, 398-409.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. New York: Addison-Wesley.

Graves, T.L. (2003a). A framework for expressing and estimating arbitrary statistical models using Markov chain Monte Carlo. Los Alamos National Laboratory Technical Report LA-UR-03-5934.

Graves, T.L. (2003b). An introduction to YADAS. `yadas.lanl.gov`.

Graves, T.L. and Hamada, M.S. (2004). Bayesian Methods for Assessing System Reliability: Models and Computation. Los Alamos National Laboratory Technical Report LA-UR-04-6834.

Hamada, M., Martz, H.F., Reese, C.S., Graves, T., Johnson, V., Wilson, A.G. (2004). A fully Bayesian approach for combining multilevel failure information in fault tree quantification and optimal follow-on

resource allocation. *Reliability Engineering and System Safety*, 86, 297-305.

Johnson, V., Graves, T., Hamada, M., Reese, C.S. (2003). A hierarchical model for estimating the reliability of complex systems (with discussion). *Bayesian Statistics 7*, Oxford University Press, 199-213, Bernardo, J.M., Bayarri, M.J., Berger, J., Dawid, A.P., Heckerman, D., Smith, A.F.M. and West, M. (Eds.).

Martz, H.F., Waller, R.A. (1990). Bayesian reliability anaysis of complex series/parallel systems of binomial subsystems and components. *Technometrics*, 32, 407-416.

Martz, H.F., Waller, R.A., Fickas, E.T. (1988). Bayesian reliability analysis of series systems of binomial subsystems and componenents. *Technometrics*, 30, 143-154.

Mastran, D.V. (1976). Incorporating component and system test data into the same assessment: a Bayesian approach. *Operations Research*, 24, 491-499.

Mastran, D.V., Singpurwalla, N.D. (1978). A Bayesian estimation of the reliability of coherent structures. *Operations Research*, 26, 663-672.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag.

Natvig, B., Eide, H. (1987). Bayesian estimation of system reliability. *Scandinavian Journal of Statistics*, 14, 319-327.

R Development Core Team (2004). R: A language and environment for statistical computing. Vienna: R Foundation for Statistical Computing. (http://www.R-project.org).

Spiegelhalter, D., Thomas, A., Best, N. (2000). *WinBUGS Version 1.3 User Manual*.